调试理论与实践

程序就是状态机，我们操作的是一个严格的数学对象。我们调试的时候就是看对应的状态有没有出错。

实际上我们的代码出问题可以分成三个部分
- Fault（设计上的错误）
- Error（产生了一个不该产生的状态）
- Failure（程序执行失败）

遇到的一个挑战就是
- 我们只能看到failue，然后去修正对应的错误。
- error到failure 可能会有一个很漫长的过程，这个导致可能只有很有经验的程序员才可能会把一些特定的error直接关联到实际的fault。

用printf（）实际上就是用输出语句，检查一些阶段的状态。（缩小可能出错的范围）
gdb调试的目的就是为了检查程序运行时的状态。（step by step）

没有经过测试的代码是错误的。
要加入assert语句，目的是尽量减少error到failure的gap

GDB是一门编程语言，本身是支持一部分自动化处理的，当你调试的时候可以，写一下对应的代码，帮你减少一些debug的痛苦。

调试理论给了大家在遇到 “任何问题” 时候 self-check 的列表：
1. 是怎样的程序（状态机）在运行？
2. 我们遇到了怎样的 failure？
3. 我们能从状态机的运行中从易到难得到什么信息？
4. 如何二分检查这些信息和 error 之间的关联？

写并发的程序，一开始就要养成一个信念，就是，我的代码其实是错的。

开头就是给我们展示了一下如何修改虚拟机内的游戏程序。红色警戒二

然后引申到编程它是一个严格的数学对象，
机器永远是对的。
这跟我们接触到的从小到大接触到的数学都有点不一样因为编程要求你这个过程中的每一个步骤都不能出错但是我们平时的数学证明题只要关键步骤对了或者说答案对了基本就不会太过追究。不会出现你只要错了一小个地方就会导致零分。

但是这个严格的数学对象也是人发明的。

三大公理
机器永远是对的
没有经过测试的程序永远是错的
任何让你觉得tedious的工作都可以被简化

我们要给我们的程序赋予对应的现实意义就比如说他可能会在代码里打空格打分段然后这样子就形成了一段一段的代码块这个事实上就跟们平时写自然语言的，。差不多

缩进有些用两个空格有些用四个空格有些用八个空格Linux里面用八个空格主要是为了让你路过这样子你进行多重嵌套的话你的代码会变得非常的Ugly然后以此来帮你不要多重嵌套

就比如说我们平时写DP的时候们的DP是什么DPI减1JJ减1I事实上我们可以给它对应的变量名增加它的可读性。
这个在二维的动态规划里面可能不是那么的明显但是如果尾数上升到五维啊六维啊那可能就非常的重要了。

当你不会的时候你可以大胆的去问gpt他看过了很多很多的代码他可能看过给了GitHub上所有的代码。

Theory
- 什么是计算

Systems
- 什么是计算机
- 如何优化计算机
- 如何制造计算机

AI
- 如何用计算机和计算实现人工智能

事实上你编程的话最好有一个人来帮你学编程，编程最好不要自学当然你的老师前提是要会编程的。
要去勇敢的面拥抱变化。比如说看一下现在的硬件长什么样子看一下gpt到底有多少**
三省吾身
滚去编程了吗?
写测试用例了吗?
回看了自己的工作流程吗?